

# 字符串 3

## 后缀数组

### 算法简介

后缀树的一种替代品，可以用于解决各种字符串问题。

### 算法实现

后缀数组的核心为  $sa$  数组、 $rk$  数组以及  $height$  数组。

其中  $sa_i$  表示排名为  $i$  的后缀的起始位置、 $rk_i$  表示起始位置为  $i$  的后缀的排名。

$sa$  数组可以通过倍增法和基数排序求解，时间复杂度  $O(n \log n)$ 。根据  $rk_{sa_i} = i$  可以  $O(n)$  求解  $rk$  数组。

$height_i$  表示  $LCP(S[sa_i, n], S[sa_{i-1}, n])$  的长度，给出引理

$$height_{rk_i} \geq height_{rk_{i-1}} - 1$$

显然

$$height_{rk_i} = LCA(S[i, n], S[sa_{rk_{i-1}}, n]), height_{rk_{i-1}} = LCP(S[i-1, n], S[sa_{rk_{i-1}-1}, n])$$

假设  $S[i-1, i-1+k] = S[j, j+k]$  显然有  $S[i, i-1+k] = S[j+1, j+k]$  于是上述引理得证。

根据引理，可以  $O(n)$  求解  $height$  数组。

同时有引理

$$LCP(S[sa_i, n], S[sa_j, n]) = \min_{i \leq k \leq j} height_k$$

可以感性理解为  $k$  指针从  $i+1$  滑到  $j$  期间  $LCP$  不增且总是取最小，于是上述结论成立。

根据该结论建立  $ST$  表即可  $O(1)$  求解每个后缀  $LCP$  的询问。

考虑如何比较任意两个子串  $S_1 = S[a, b], S_2 = S[c, d]$  的大小。

若  $LCP(S[a, n], S[b, n]) \geq \min(S_1, S_2)$  显然只需要比较  $|S_1|, |S_2|$  否则比较  $rk_a, rk_c$  即可。

下面给出后缀数组模板。

```
namespace SA{
    int sa[MAXN], rk[MAXN], height[MAXN], X[MAXN], Y[MAXN], c[MAXN];
    int d[MAXN][MAXM], lg2[MAXN];
    void get_sa(int *s, int n, int m){ // s 下标从1开始
        int *x=X, *y=Y;
        _rep(i, 0, m) c[i]=0;
```

```
_rep(i,1,n)c[x[i]=s[i]]++;
_rep(i,1,m)c[i]+=c[i-1];
for(int i=n;i;i--)sa[c[x[i]]--]=i;
for(int k=1;k<n;k<=<=1){
    int pos=0;
    _rep(i,n-k+1,n)y[++pos]=i;
    _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
    _rep(i,0,m)c[i]=0;
    _rep(i,1,n)c[x[i]]++;
    _rep(i,1,m)c[i]+=c[i-1];
    for(int i=n;i;i--)sa[c[x[y[i]]]]--=y[i],y[i]=0;
    swap(x,y);
    pos=0,y[n+1]=0;
_rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
    if(pos==n)break;
    m=pos;
}
}
void get_height(char *s,int n){//必须先得到sa数组和rk数组
    for(int i=1,k=0;i<=n;i++){
        if(k)k--;
        while(s[i+k]==s[sa[rk[i]-1]+k])k++;
        height[rk[i]]=k;
    }
}
void build_st(int n){
    lg2[1]=0;
    _rep(i,2,n)
    lg2[i]=lg2[i>>1]+1;
    _rep(i,1,n)
    d[i][0]=height[i];
    for(int j=1;(1<<j)<=n;j++){
        _rep(i,1,n+1-(1<<j))
        d[i][j]=min(d[i][j-1],d[i+(1<<(j-1))][j-1]);
    }
}
int lcp(int a,int b){//a!=b
    int lef=rk[a],rig=rk[b];
    if(lef>rig)swap(lef,rig);lef++;
    int len=rig-lef+1;
    return min(d[lef][lg2[len]],d[rig-(1<<lg2[len])+1][lg2[len]]);
}
}
```

## 算法例题

## 例题一

## 洛谷p4051

## 题意

给定字符串  $S=s_1s_2\cdots s_n$  将其视为一个环，任意选择环的起点，可以得到  $n$  个新字符串  $T_k=s_{k+1}\cdots s_{k-1}$

询问将所有  $T_i$  按字典序从小到大排序后依次取每个  $T_i$  的最后一个字母构成的字符串。

## 题解

考虑将  $S$  倍长为  $SS$  求  $SS$  每个后缀的排名，即可得到每个字符串  $T_i$  的排名。

关于正确性，考虑字符串  $abc$  于是  $T_2$  代表的字符串  $bca$  变为  $bcabc$  实际上这相当于  $T_2$  再与  $T_2$  的前缀拼接而成，不影响排序结果。

时间复杂度  $O(n\log n)$

```

const int MAXN=2e5+5;
namespace SA{
    int sa[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
    }
}
char buf[MAXN];
int main()

```

```
{
    scanf("%s",buf+1);
    int n=strlen(buf+1);
    _rep(i,1,n)buf[i+n]=buf[i];
    buf[2*n+1]='\0';
    SA::get_sa(buf,n<<1,'z');
    _rep(i,1,2*n){
        if(SA::sa[i]<=n)
            putchar(buf[SA::sa[i]+n-1]);
    }
    return 0;
}
```

## 例题二

### 洛谷p2870

#### 题意

给定一个字符串  $S$  和一个空串  $T$  每个可以选择  $S$  的首字符或末字符，将其删去后加入到  $T$  末尾。  
问所有可能的  $T$  中字典序最小的。

#### 题解

考虑贪心，假设现在字符串为  $s_L s_{L+1} \dots s_{R-1} s_R$  显然选取  $s_L, s_R$  中字典序最小的最优。

如果  $s_L = s_R$  接下来考虑选择  $s_{L+1}, s_{R-1}$  中字典序最小的，直到比较到端点为止。

上述操作等价于比较  $S[L,n]$  和  $S[1,R]$  的字典序。考虑构造字符串  $s_1 s_2 \dots s_n + \text{\texttt{\0}} + s_n \dots s_2 s_1$

于是可以通过后缀数组得到  $S[L,n]$  和  $S[1,R]$  的排名，时间复杂度  $O(n \log n)$

```
const int MAXN=1e6+5;
namespace SA{
    int sa[MAXN],rk[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
        }
    }
}
```

```

        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
        swap(x,y);
        pos=0,y[n+1]=0;
        _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++
pos;
        if(pos==n)break;
        m=pos;
    }
    _rep(i,1,n)rk[sa[i]]=i;
}
}
char buf[MAXN];
int main()
{
    int n=read_int(),len=2*n+1;
    _rep(i,1,n)buf[i]=buf[len+1-i]=get_char();
    buf[n+1]=0;
    SA::get_sa(buf,len,'Z');
    int L=1,R=n,cnt=0;
    while(L<=R){
        if(SA::rk[L]<SA::rk[len+1-R])
            putchar(buf[L++]);
        else
            putchar(buf[R--]);
        if(++cnt%80==0)putchar('\n');
    }
    return 0;
}

```

### 例题三

[洛谷p2408](#)

#### 题意

给定一个字符串  $S$  求本质不同的子串个数。

#### 题解

子串相当于某个后缀的前缀，于是考虑依次枚举  $sa_i$  代表的后缀，加上新增的本质不同的前缀。

根据  $LCP$  性质不难得到属于  $sa_i$  后缀的新增的本质不同于  $sa_1 \dots sa_{i-1}$  的所有前缀的子串数为  $n+1-sa_i$

于是最终答案为  $\sum_{i=1}^n n+1-sa_i-\text{height}_i=\frac{n(n+1)}{2}-\sum_{i=1}^n \frac{n(n+1)}{2}-\sum_{i=2}^n \text{height}_i$

```
const int MAXN=1e5+5;
namespace SA{
    int sa[MAXN],rk[MAXN],height[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&& y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
    void get_height(char *s,int n){
        for(int i=1,k=0;i<=n;i++){
            if(k)k--;
            while(s[i+k]==s[sa[rk[i]-1]+k])k++;
            height[rk[i]]=k;
        }
    }
}
char buf[MAXN];
int main()
{
    int n=read_int();
    scanf("%s",buf+1);
    SA::get_sa(buf,n,'z');
    SA::get_height(buf,n);
    LL ans=1LL*n*(n+1)/2;
    _rep(i,2,n)ans-=SA::height[i];
    enter(ans);
    return 0;
}
```

## 例题四

## 洛谷p2852

## 题意

给定一个字符串  $S$  求至少出现  $k$  次的最长子串的长度。

## 题解

考虑至少出现  $k$  次的子串，他一定是至少连续  $k-1$  个  $\text{height}$  数组代表的  $k$  个后缀的公共前缀。

于是求出  $\text{height}$  数组后单调队列维护每个长度为  $k-1$  的连续区间的  $\text{height}$  数组的最小值的最大值即可。

时间复杂度  $O(n \log n)$

```

const int MAXN=2e4+5;
namespace SA{
    int sa[MAXN],rk[MAXN],height[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(int *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
    void get_height(int *s,int n){
        for(int i=1,k=0;i<=n;i++){
            if(k)k--;
            while(s[i+k]==s[sa[rk[i]-1]+k])k++;
            height[rk[i]]=k;
        }
    }
}

```

```
    }  
  }  
}  
int a[MAXN], b[MAXN], q[MAXN];  
int main()  
{  
  int n=read_int(), k=read_int()-1;  
  _rep(i, 1, n) a[i]=b[i]=read_int();  
  sort(b+1, b+n+1);  
  int m=unique(b+1, b+n+1)-b;  
  _rep(i, 1, n) a[i]=lower_bound(b+1, b+m, a[i])-b;  
  SA::get_sa(a, n, m);  
  SA::get_height(a, n);  
  int ans=0, tail=1, head=0;  
  _rep(i, 1, n){  
    while(tail<=head&& i-q[tail]>=k) tail++;  
    while(tail<=head&& SA::height[i]<=SA::height[q[head]]) head--;  
    q[++head]=i;  
    if(i>=k) ans=max(ans, SA::height[q[tail]]);  
  }  
  enter(ans);  
  return 0;  
}
```

## 例题五

[洛谷p2178](#)

### 题意

给定一个字符串  $S$  和序列  $v$  字符串对  $(S[a,b], S[c,d])$  的权值为  $v_{a-b}$

对  $0 \leq i \leq n$  询问满足  $S[a, a+i-1] = S[b, b+i-1]$  的所有字符串对的个数和最大权值。

### 题解

枚举子串的右端点，设  $\text{LCP}(S[sa_a, n], S[sa_b, n]) = k$  显然  $S[sa_a, sa_b+i-1] = S[sa_b, sa_b+i-1] (1 \leq i \leq k)$  均成立。

不妨只考虑字符串  $S[sa_a, sa_a+k-1] = S[sa_b, sa_b+k-1]$  对  $i=k$  的答案的贡献，最后求后缀和以及后缀最大值即可。

将  $\text{height}_i$  视为连接  $sa_i$  和  $sa_{i-1}$  的一条边，将边按  $\text{height}$  值从大到小排序。

然后依次向图中加入每条边，于是每条新加入的边一定是最短边，决定了两个连通块间的  $\text{LCP}$

于是并查集维护点集大小，点集中的  $sv$  的最大值和最小值(因为可能出现负数乘以负数的情况) 即可。

时间复杂度  $O(n \log n)$

```

const int MAXN=3e5+5;
namespace SA{
    int sa[MAXN],rk[MAXN],height[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&& y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
    void get_height(char *s,int n){
        for(int i=1,k=0;i<=n;i++){
            if(k)k--;
            while(s[i+k]==s[sa[rk[i]-1]+k])k++;
            height[rk[i]]=k;
        }
    }
}
char buf[MAXN];
int p[MAXN],sz[MAXN],maxv[MAXN],minv[MAXN];
LL ans1[MAXN],ans2[MAXN];
pair<int,int> edge[MAXN];
int Find(int x){return x==p[x]?x:p[x]=Find(p[x]);}
bool cmp(const pair<int,int> &a,const pair<int,int> &b){
    return a.first>b.first;
}
int main()
{
    int n=read_int();
    scanf("%s",buf+1);
    SA::get_sa(buf,n,'z');
}

```

```
SA::get_height(buf,n);
_rep(i,1,n){
    maxv[i]=minv[i]=read_int();
    p[i]=i,sz[i]=1;
}
_for(i,0,n)ans2[i]=-1e18;
_for(i,1,n)edge[i]=make_pair(SA::height[i+1],i+1);
sort(edge+1,edge+n,cmp);
_for(i,1,n){
    int
u=SA::sa[edge[i].second],v=SA::sa[edge[i].second-1],k=edge[i].first;
    int x=Find(u),y=Find(v);
    ans1[k]+=1LL*sz[x]*sz[y];
    ans2[k]=max(ans2[k],max(1LL*maxv[x]*maxv[y],1LL*minv[x]*minv[y]));
    p[x]=y;
    sz[y]+=sz[x];
    maxv[y]=max(maxv[y],maxv[x]);
    minv[y]=min(minv[y],minv[x]);
}
for(int i=n-1;i;i--){
    ans1[i-1]+=ans1[i];
    ans2[i-1]=max(ans2[i-1],ans2[i]);
}
_for(i,0,n){
    space(ans1[i]);
    enter(ans1[i]?ans2[i]:0);
}
return 0;
}
```

## 例题六

### 洛谷p3763

#### 题意

给定一个文本串  $S_0$  和模式串  $S$  问  $S_0$  中  $S$  的近似匹配次数。

定义  $S_0[i,i+n-1]$  与  $S[1,n]$  近似匹配当且仅当  $S_0[i,i+n-1]$  与  $S[1,n]$  至多有  $3$  个不同的字符。

#### 题解

由于匹配最多只允许  $3$  个字符不同，考虑暴力枚举  $S_0$  右端点，然后尝试匹配。

将串拼接为  $S_0+\text{\#}+S$  即可  $O(1)$  完成相应  $\text{\{LCP\}}$  查询，然后每次跳  $\text{\{LCP\}}$  的长度继续匹配即可。

时间复杂度  $O(n \log n)$  主要在于后缀数组和  $\text{ST}$  表建立，常数较大。

ps.这题也有  $\text{FFT}$  的解法，有兴趣的可以查看这篇[题解](#)

```

const int MAXN=2e5+5,MAXM=20;
namespace SA{
    int sa[MAXN],rk[MAXN],height[MAXN],x[MAXN],y[MAXN],c[MAXN];
    int d[MAXN][MAXM],lg2[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
    void get_height(char *s,int n){
        for(int i=1,k=0;i<=n;i++){
            if(k)k--;
            while(s[i+k]==s[sa[rk[i]-1]+k])k++;
            height[rk[i]]=k;
        }
    }
    void build_st(int n){
        lg2[1]=0;
        _rep(i,2,n)
        lg2[i]=lg2[i>>1]+1;
        _rep(i,1,n)
        d[i][0]=height[i];
        for(int j=1;(1<<j)<=n;j++){
            _rep(i,1,n+1-(1<<j))
            d[i][j]=min(d[i][j-1],d[i+(1<<(j-1))][j-1]);
        }
    }
    int lcp(int a,int b){
        int lef=rk[a],rig=rk[b];
    }
}

```

```
        if(lef>rig)swap(lef,rig);lef++;
        int len=rig-lef+1;
        return min(d[lef][lg2[len]],d[rig-(1<<lg2[len])+1][lg2[len]]);
    }
}
char buf[MAXN];
int main()
{
    int T=read_int();
    while(T--){
        scanf("%s",buf+1);
        int len1=strlen(buf+1);
        buf[len1+1]='#';
        scanf("%s",buf+len1+2);
        int n=strlen(buf+1),len2=n-len1-1;
        SA::get_sa(buf,n,'z');
        SA::get_height(buf,n);
        SA::build_st(n);
        int ans=0;
        _rep(i,1,len1-len2+1){
            int pos=1,cnt=0;
            while(cnt<=3){
                pos+=SA::lcp(i+pos-1,len1+1+pos);
                if(pos>len2)break;
                pos++;cnt++;
            }
            if(cnt<=3)
                ans++;
        }
        enter(ans);
    }
    return 0;
}
```

## 例题七

[洛谷p4248](#)

### 题意

给定字符串  $S$  设  $T_i = S[i, n]$  求

$$\sum_{1 \leq i < j \leq n} |T_i| + |T_j| + 2 \times \text{LCP}(T_i, T_j)$$

### 题解

$$\sum_{1 \leq i < j \leq n} |T_i| + |T_j| - 2 \times \text{LCP}(T_i, T_j) = (n-1) \sum_{i=1}^n |T_i| - 2 \times \sum_{1 \leq i < j \leq n} \text{LCP}(T_i, T_j) = \frac{(n-1)n(n+1)}{2} - 2 \times \sum_{1 \leq i < j \leq n} \text{LCP}(T_i, T_j)$$

发现只要维护后面部分即可。

考虑单调栈维护，设  $\text{dp}_i = \sum_{1 \leq j < i} \text{LCP}(T_i, T_j)$  用单调栈维护  $\text{height}_i$  数组。

于是对所有  $i$  可以均摊  $O(1)$  得到最大的  $j < i$  满足  $\text{height}_j \leq \text{height}_i$

根据  $\text{height}$  性质，发现  $j < k < i$  时  $\text{LCP}(T_k, T_i) = \text{height}_i$  而  $k < j$  时  $\text{LCP}(T_k, T_i) = \text{LCP}(T_k, T_j)$

于是  $\text{dp}_i = \sum_{1 \leq j < i} \text{LCP}(T_i, T_j) = \text{dp}_j + (i-j) \times \text{height}_i$

另外该题也可以用笛卡尔树或并查集等维护。


```

const int MAXN=5e5+5,MAXM=20;
namespace SA{
    int sa[MAXN],rk[MAXN],height[MAXN],x[MAXN],y[MAXN],c[MAXN];
    void get_sa(char *s,int n,int m){
        _rep(i,0,m)c[i]=0;
        _rep(i,1,n)c[x[i]=s[i]]++;
        _rep(i,1,m)c[i]+=c[i-1];
        for(int i=n;i;i--)sa[c[x[i]]--]=i;
        for(int k=1;k<n;k<=<=1){
            int pos=0;
            _rep(i,n-k+1,n)y[++pos]=i;
            _rep(i,1,n)if(sa[i]>k)y[++pos]=sa[i]-k;
            _rep(i,0,m)c[i]=0;
            _rep(i,1,n)c[x[i]]++;
            _rep(i,1,m)c[i]+=c[i-1];
            for(int i=n;i;i--)sa[c[x[y[i]]]--]=y[i],y[i]=0;
            swap(x,y);
            pos=0,y[n+1]=0;
            _rep(i,1,n)x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&& y[sa[i]+k]==y[sa[i-1]+k])?pos:++pos;
            if(pos==n)break;
            m=pos;
        }
        _rep(i,1,n)rk[sa[i]]=i;
    }
    void get_height(char *s,int n){
        for(int i=1,k=0;i<=n;i++){
            if(k)k--;
            while(s[i+k]==s[sa[rk[i]-1]+k])k++;
            height[rk[i]]=k;
        }
    }
}
char buf[MAXN];

```

```
pair<int,int> Stack[MAXN];
LL dp[MAXN];
int main()
{
    scanf("%s",buf+1);
    int n=strlen(buf+1);
    SA::get_sa(buf,n,'z');
    SA::get_height(buf,n);
    LL ans=1LL*(n-1)*n*(n+1)/2;
    int top=0;
    Stack[0]=make_pair(0,0);
    _rep(i,1,n){
        while(top&&Stack[top].second>SA::height[i])top--;
        dp[i]=dp[Stack[top].first]+1LL*(i-
Stack[top].first)*SA::height[i],ans-=dp[i]*2;
        Stack[++top]=make_pair(i,SA::height[i]);
    }
    enter(ans);
    return 0;
}
```

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:%E5%AD%97%E7%AC%A6%E4%B8%B2\\_3&rev=1598869056](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%AD%97%E7%AC%A6%E4%B8%B2_3&rev=1598869056) 

Last update: 2020/08/31 18:17